
RigAnyFace: Scaling Neural Facial Mesh Auto-Rigging with Unlabeled Data

Appendix

Anonymous Author(s)

Affiliation

Address

email

1 A Training and Inference Details

2 In the first stage of training, the weights for the image loss, mask loss, 2D displacement loss, and
3 regularization loss are set to 10.0, 1.0, 1.0, and 0.0001, respectively. In the second stage, the weights
4 for the image loss, mask loss, 3D MSE loss, 2D landmark loss, and 2D eye closure loss are set to
5 10.0, 1.0, 100.0, 0.5, and 0.5, respectively. We train our model on an Nvidia A100 instance with 8
6 GPUs and a total batch size of 8 (i.e., effectively 1 sample per GPU if using distributed data parallel).
7 The training proceeds in two stages. For the first stage, we train the deformation model on both
8 rigged and unrigged head datasets (8,386 samples in total) using only 2D supervision for 15 epochs.
9 This stage typically takes around 1.5 days to complete. For the second stage, we then finetune the
10 model from the first stage on the rigged head dataset (2,929 samples), incorporating both 2D and
11 3D supervision for 20 epochs. This finetuning phase finishes in approximately 1 day. Throughout
12 both stages, we use the Adam optimizer, initializing the learning rate at 0.0001. For learning rate
13 scheduling, we employ CosineAnnealingWarmRestarts, allowing it to decay from 0.0001 to nearly
14 0 by the end of training. Additionally, we use a warm-up phase of 20,000 steps to stabilize early
15 training.

16 For inference speed, our model runs a single forward pass to predict blendshapes offline, requiring
17 only one run per input mesh. The outputs are converted into classical FACS blendshape rigs, enabling
18 efficient animation by simply linear blending. The proposed model consists of 5.4M parameters and
19 it takes on average 8.72s on an Apple M2 Max CPU and 3.1s on a Nvidia T4 GPU to generate a
20 FACS blendshape rig on the test set (1,750 vertices, 3,362 faces on average).

21 B Details for 2D Displacement Calculation

22 In the following code sample, we demonstrate how to compute the 2D displacement of each pixel
23 from mesh vertex deformations in a fully differentiable manner. This implementation leverages
24 PyTorch3D's differentiable rendering functionality.

```
25 def render_displacement(vertices, deformed_vertices, faces, renderer, camera, res
26     =(512,512)):
27     """
28     Parameters
29     -----
30     vertices: torch.tensor (V, 3)
31     deformed_vertices: torch.tensor (V, 3)
32     faces: torch.tensor (F, 3)
33     renderer: pytorch3d.renderer.MeshRenderer object
34     camera: pytorch3d.renderer.cameras.CamerasBase object
35     res: tuple
```

```

36
37 Returns
38 -----
39 displacement_2D: torch.tensor (res[0], res[1], 2)
40 """
41
42 verts_2d = camera.transform_points_screen(vertices, image_size=res)
43 verts_2d_deformed = camera.transform_points_screen(deformed_vertices, image_size
44 =res)
45 verts_flow = (verts_2d_deformed - verts_2d)[: , :2] # Vx2
46 verts_flow = verts_flow / res * 0.5 + 0.5 # 0~1
47 flow_tex = torch.nn.functional.pad(verts_flow, pad=[0, 1]) # Vx3
48 texture = TexturesVertex(verts_features=[flow_tex])
49 meshes = pytorch3d.structures.Meshes(
50     verts=[vertices], faces=[faces], textures=texture
51 )
52 displacement_2D = renderer(meshes, cameras=camera)
53
54 return displacement_2D[... , :2].squeeze()

```

55 C Effectiveness of 2D Generation Pipeline

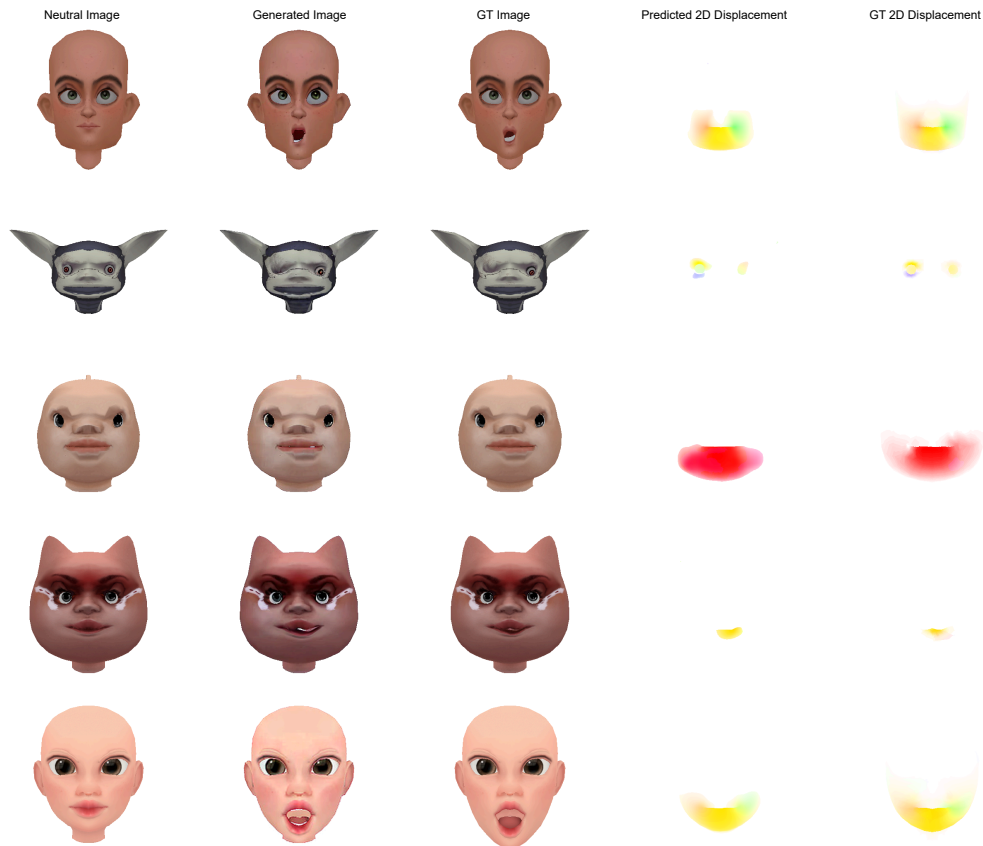


Figure 1: Example results of 2D generation pipeline.

56 To validate the effectiveness of our 2D supervision generation pipeline, we exclude several rigged
57 heads during the fine-tuning of the 2D face animation model and the flow estimation model. In Fig. 1,
58 we present random sample results showcasing different faces and poses. The ground truth images
59 and 2D displacements are rendered using the ground truth deformations of the rigged heads. The

2D face animation model generates pose images based on the neutral image input, while the flow estimation model takes the neutral and generated images as input to predict the 2D displacement. The 2D displacement is visualized according to the standard optical flow convention.

D Data Collection Details

FACS Poses For each rigged heads, our artist annotated 48 FACS poses and 48 corrective poses as blendshape rig. We show those 48 FACS poses in Tab. 1, and Fig 2. In addition to blendshapes for individual FACS poses, we generate corrective blendshapes by linearly combining certain poses and manually correcting artifacts. These corrective blendshapes account for the complex deformations resulting from pose interactions.

Semantic Annotation We provide a semantic annotation map for rigged heads, labeling different regions on the mesh (e.g., ears, mouth, eyes), along with facial landmark annotations specified as vertex indices. These annotations allow for the application of weighted losses or region-specific training objectives.

Head Interpolation First, we standardized the UV layout across all head meshes, ensuring that corresponding facial features like eyes and mouths occupy the same region in UV space. This consistent mapping enables the identification of 3D correspondences between vertices on different meshes. Using these correspondences, we can smoothly interpolate between different head geometries through linear blending to significantly increases the size of our dataset.

#	SHORT	FULL	#	SHORT	FULL
1	neutral	neutral	25	l_EC	LeftEyeClosed
2	c_COR	Corrugator	26	l_EULR	LeftEyeUpperLidRaiser
3	c_CR	ChinRaiser	27	l_IBR	LeftInnerBrowRaiser
4	c_CRUL	ChinRaiserUpperLip	28	l_LCD	LeftLipCornerDown
5	c_ELD	EyesLookDown	29	l_LCP	LeftLipCornerPuller
6	c_ELL	EyesLookLeft	30	l_LLD	LeftLowerLipDepressor
7	c_ELR	EyesLookRight	31	l_LS	LeftLipStretcher
8	c_ELU	EyesLookUp	32	l_NW	LeftNoseWrinkler
9	c_FN	Funneler	33	l_OBR	LeftOuterBrowRaiser
10	c_FP	FlatPucker	34	l_ULR	LeftUpperLipRaiser
11	c_JD	JawDrop	35	r_BL	RightBrowLowerer
12	c_JL	JawLeft	36	r_CHP	RightCheekPuff
13	c_JR	JawRight	37	r_CHR	RightCheekRaiser
14	c_LLS	LowerLipSuck	38	r_DM	RightDimpler
15	c_LP	LipPresser	39	r_EC	RightEyeClosed
16	c_LPT	LipsTogether	40	r_EULR	RightEyeUpperLidRaiser
17	c_ML	MouthLeft	41	r_IBR	RightInnerBrowRaiser
18	c_MR	MouthRight	42	r_LCD	RightLipCornerDown
19	c_PK	Pucker	43	r_LCP	RightLipCornerPuller
20	c_ULS	UpperLipSuck	44	r_LLD	RightLowerLipDepressor
21	l_BL	LeftBrowLowerer	45	r_LS	RightLipStretcher
22	l_CHP	LeftCheekPuff	46	r_NW	RightNoseWrinkler
23	l_CHR	LeftCheekRaiser	47	r_OBR	RightOuterBrowRaiser
24	l_DM	LeftDimpler	48	r_ULR	RightUpperLipRaiser

Table 1: FACS Short and Full Name Mapping.

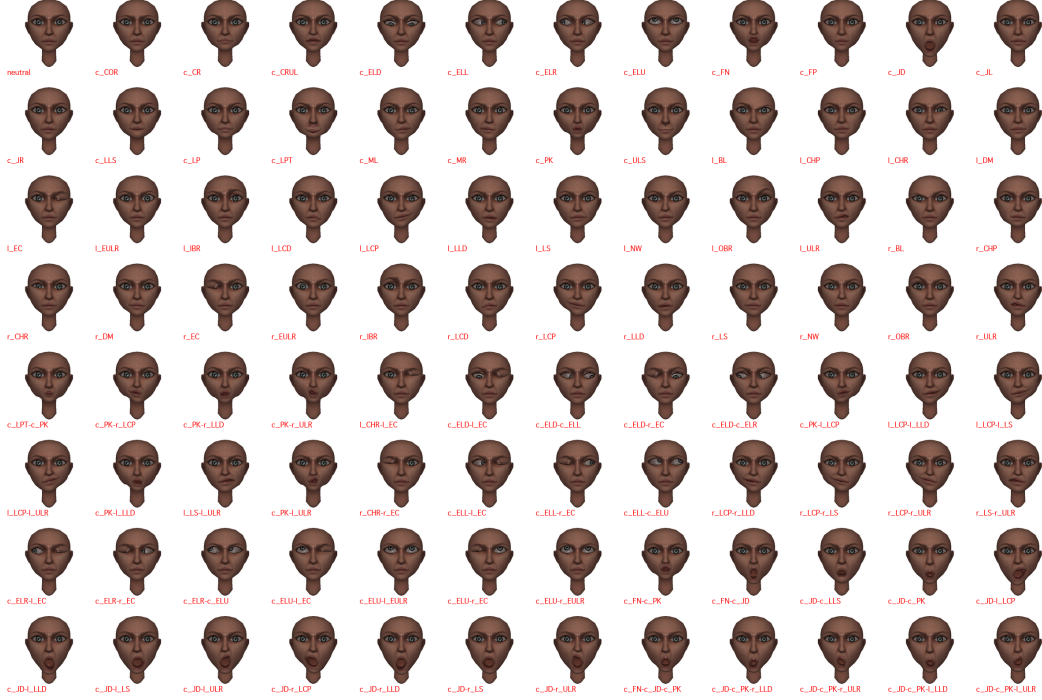


Figure 2: A sample of all the FACS and corrective poses used in this work.

78 E Dataset Split

79 Our dataset includes 161 rigged heads and 175 unriggered heads. From these, a subset of 24 rigged
80 heads with 3D ground-truth annotations forms the test set to for accurate absolute error evaluation.
81 Additionally, we select 37 diverse unriggered heads as the test set, representing different species and
82 shapes to evaluate the model’s generalization on out-of-distribution (OOD) faces. For training, we
83 augment the dataset using interpolations, manually filtering out poor interpolation results. Specifically,
84 we interpolate the remaining 137 unriggered heads with a factor of 50, generating 5,457 samples, and
85 interpolate the remaining 137 rigged heads with a factor of 25, producing 2,929 samples.

86 F Pre-processing for Baseline Method NFR

87 All NFR baseline results were obtained after applying the official preprocessing pipeline¹: we keep
88 only the largest connected component and remove the inner-lip and eyelid surfaces. These steps
89 are crucial for NFR to generate reasonable deformations. Figure 3 shows that retaining multiple
90 disconnected components causes self-penetration, while Fig 4 shows jarring artifacts when the
91 inner-lip surfaces are not trimmed. In contrast, our method do not need such preprocessing.

92 G Border Impact

93 Our face-autorigging framework could broaden access to high-quality animation by letting small
94 studios, educators, and assistive-tech developers create expressive avatars quickly, which benefits
95 entertainment, remote communication, and certain medical visualization tasks. However, the same
96 ease of use can lower the barrier for deepfake production, intensifying privacy concerns around
97 emotion tracking and biometric profiling. Careful dataset curation, explicit usage licenses, and
98 watermarking tools are essential to realize the creative upside while limiting misuse and inequitable
99 impacts.

¹https://github.com/dafei-qin/NFR_pytorch

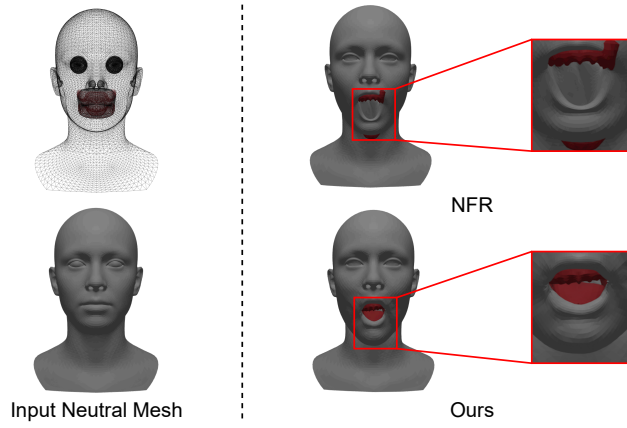


Figure 3: Compared to NFR during inference on meshes with multiple disconnected components from ICT Facekit Dataset. We highlight one of these components: "gums and tongue" in red. While animating a Jaw Drop pose, this component causes penetration issues for NFR.

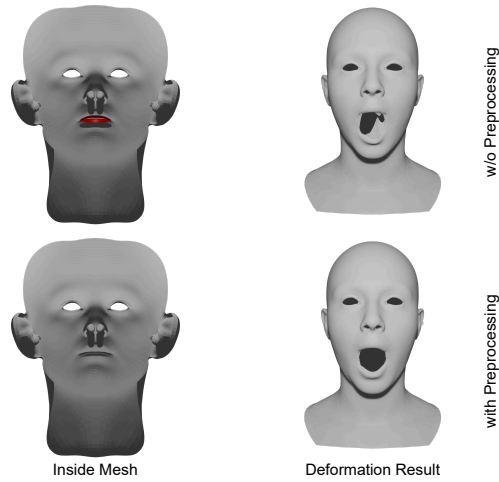


Figure 4: Illustration of the trimming preprocessing step for NFR. The inner-lip surfaces to be trimmed are highlighted in red in the top-left figure. Omitting this step results in implausible deformations produced by NFR.